What is claimed is:

1. A method of estimating query progress, comprising:

a) defining a model of work performed during execution of a query;

b) estimating a total amount of work that will be performed according to the model during execution of the query;

c) estimating an amount of work performed according to the model at a given point during the execution of the query; and

d) estimating the progress of the query using the amount of work performed and the total amount of work.

2. The method of claim 1 further comprising displaying estimated progress of the query to a user.

3. The method of claim 1 wherein work performed during execution of a query is modeled as a number items returned by a query operator.

4. The method of claim 1 wherein work performed during execution of a query is modeled as a number of GetNext() calls by a query operator.

5. The method of claim 1 wherein the work performed during execution of the query is modeled as work performed by a driver node operator during execution of the query.

6. The method of claim 1 wherein work performed by a driver node operator is modeled as a number of items returned by the driver node operator.

7. The method of claim 1 wherein work performed by a driver node operator is modeled as a number of GetNext() calls by a driver node operator.

8. The method of claim 1 further comprising dividing a query execution plan into a set of pipelines and estimating the progress of each pipeline.

9. The method of claim 8 wherein the pipelines comprise sequences of non-blocking operators.

10. The method of claim 8 further comprising combining progress estimates for the pipelines to estimate the progress of the query.

11. The method of claim 8 further comprising initializing an estimate of the total amount of work that will be performed by a pipeline with an estimate from a query optimizer.

12. The method of claim 1 further comprising refining the initial estimate of the total work using feedback obtained during query execution.

13. The method of claim 8 further comprising identifying driver node operators of the pipeline and modeling the work performed during execution of the pipelines as work performed by the driver node operators.

14. The method of claim 8 further comprising modeling the work performed during execution of the pipelines as work performed by all operators in the pipeline.

15. The method of claim 8 further comprising identifying driver node operators of the pipeline and using information about the driver node operators obtained during execution to estimate a total amount of work that will be performed by all operators in the pipeline.

16. The method of claim 2 further comprising preventing decreasing progress estimations from being displayed to the user.

17. The method of claim 16 wherein decreasing progress estimations are prevented by using an upper bound on the total work that will be performed as an estimate of the total work that will be performed.

18. The method of claim 1 further comprising identifying a spill of tuples during query execution and adjusting the model of work to account for additional work that results from the spill of tuples.

19. The method of claim 10 further comprising assigning weights to the pipelines.

20. The method of claim 19 wherein the weights are based on relative execution rates of the pipelines.

21. The method of claim 1 further comprising updating an estimated total amount of work that will be performed during query execution.

22. The method of claim 1 wherein an estimated amount of work performed according to the model is updated at a plurality of points during query execution.

23. The method of claim 1 further comprising maintaining an upper bound and a lower bound on the on the total amount of work that will be performed and modifying an estimated total amount of work that will be performed when the estimated total amount of work that will be performed is outside a range defined by the upper bound and the lower bound.

24. The method of claim 3 further comprising maintaining an upper bound and a lower bound on the on a total number of items that will be returned by the query operator and modifying an estimated total number of items that will be returned by the query operator when the estimated total number of items that will

be returned by the query operator is outside a range defined by the upper bound and the lower bound.

25. The method of claim 24 wherein a rule used for maintaining a bound on the total number of items that will be returned by the query operator is specific to the query operator.

26. The method of claim 25 wherein the query operator is a Group By operator and the rule used for maintaining an upper bound on a number of groups that will be returned by the Group By operator comprises subtracting a number of items returned by an immediately preceding operator in a query execution plan from an upper bound of the immediately preceding operator and adding a number of distinct values observed by the Group By operator.

27. The method of claim 25 wherein the query operator is a Hash Join operator and the rule used for maintaining an upper bound on the number of rows that will be returned by the Hash Join operator comprises subtracting a number of items returned by an immediately preceding operator in a query execution plan from an upper bound of the immediately preceding operator and multiplying a number of rows of a largest build partition.

28. The method of claim 24 further comprising setting the lower bound to a number of items returned by the query operator at a given point during query execution.

29. The method of claim 24 wherein the upper bound of the query operator is maintained using an upper bound of one or more preceding query operators in a query execution plan.

30. The method of claim 29 wherein the upper bound of the query operator is maintained using an upper bound of an immediately preceding query operator in the query execution plan.

31. The method of claim 24 wherein the upper bound of the query operator is maintained using a number of items returned by one or more preceding operators in a query execution plan at a given point during query execution.

32. The method of claim 31 wherein the upper bound of the query operator is maintained using a number of items returned by an immediately preceding query operator in the query execution plan.

33. The method of claim 24 wherein the upper bound of the query operator is maintained using a number of items returned by the query operator at a given point during query execution.

34. The method of claim 24 wherein upper and lower bounds are maintained for a plurality of query operators in a query execution plan and wherein a changes in bounds of query operators are periodically propagated to other query operators in the query execution plan.

35. Computer readable media comprising computer-executable instructions for performing the method of claim 1

36. In a computer system including a display, a user input facility, and an application for presenting a user interface on the display, a user interface comprising:

a) a query progress indicator that provides an indication to a user of an execution state of a query; and

b) a query end selector that allows the user to abort execution of the query.

37. The user interface of claim 36 wherein the query progress indicator provides a visual indication of a percentage of query execution that has been completed.

38. The user interface of claim 37 wherein the percentage of query execution that has been completed is estimated by dividing a number of tuples returned by the query by an estimated total number of tuples to be returned by the query.

39. The user interface of claim 37 wherein the percentage of query execution that has been completed is estimated by dividing a number of tuples returned by an operator by an estimated total number of tuples to be returned by the operator.

40. The user interface of claim 37 wherein the percentage of query execution that has been completed is estimated by dividing a GetNext() calls by a query operator by an estimated total number of GetNext() calls by the operator.

41. The user interface of claim 37 further comprising initializing the estimated total number of GetNext() calls with an estimate from a query optimizer.

42. The user interface of claim 41 wherein initial estimate of the total number of GetNext() calls is updated using feedback obtained during query execution.

43. The user interface of claim 36 the query progress indicator is prevented from providing an indication of decreasing query progress.

44. The user interface of claim 36 further comprising a tuple spill indicator that alerts a user when tuples spill to disk during query execution.

45. A system for providing an indication of query progress, comprising:

a) a user input device enabling a user to begin execution of a query and abort execution of a query;

b) a display;

c) a data content that queries can be executed upon;

d) a memory in which machine instructions are stored;

e) a processor that is coupled to the user input device, to the display, to the data content, and to the memory, the processor executing the machine instructions to carry out a plurality of functions, including:

i) executing a query upon the data content;

ii) monitoring progress of the query; and

iii) providing an indicator of query progress on the display.

46. The system of claim 45 wherein query progress is estimated as an amount of work performed at a current point of query execution divided by an estimated total amount of work.

47. The system of claim 45 wherein the processor identifies a spill of tuples during query execution and provides an indication of the spill on the display.

48. The system of claim 45 the indicator of query progress provides a visual indication of a percentage of query execution that has been completed.